

Algorithmen und Datenstrukturen

Aufgabe 1 **Validation**

- a) Die Funktion **Multiply**(a, b), die das Produkt $a \cdot b$ berechnet, sei implementiert mit vorzeichenbehafteten (signed) Variablen von 8 Bit Länge. Geben Sie Extremfälle für die Validation an!
- b) Für einen White-Box-Test sei die Funktion implementiert wie aus einer früheren Aufgabe bekannt:

```
Multiply( $a, b$ ):  
    if ( $a == 0 \vee b == 0$ ) {  
         $c = 0$ ;  
    }  
    else if ( $a < 0$ ) {  
         $c = -\text{Multiply}(-a, b)$ ;  
    }  
    else {  
         $c = 0$ ;  
         $i = 1$ ;  
        while ( $i \leq a$ ) {  
             $c = c + b$ ;  
             $i = i + 1$ ;  
        }  
    }  
    return  $c$ ;
```

Geben Sie Testfälle an, so dass jeder Fall der bedingten Anweisung mindestens einmal, Schleifen sogar mindestens zweimal durchlaufen werden!

- c) Gegeben sei ein Sortieralgorithmus. Kann in einem Black-Box-Test bestimmt werden, nach welchem Verfahren sortiert wird?

Aufgabe 2 Wachstumsraten

Identifizieren Sie im angegebenen Diagramm die Kurven der Funktionen $f(n) = 1$, $f(n) = \log(n)$, $f(n) = n$, $f(n) = n \log(n)$, $f(n) = n^2$ und $f(n) = 2^n$!



Aufgabe 3 Rechnen mit Landau-Symbolen

Zeigen Sie folgende Behauptungen:

- a) $7n^4 = O(n^5)$
- b) $n^2/2 - 2n = \Theta(n^2)$
- c) $2^{n+1} = O(2^n)$
- d) $2^{2n} \neq O(2^n)$

Aufgabe 4 Rechnen mit Landau-Symbolen

Treffen folgende Behauptungen zu?

- a) Die Worst-Case-Komplexität eines Algorithmus sei $\Theta(n)$. Nachdem auch $2n = \Theta(n)$, entspricht die Dauer eines Worst-Case-Beispiels der Länge $2n$ ungefähr der Dauer eines Worst-Case-Beispiels der Länge n .
- b) Sei die Komplexität einer Funktion f bestimmt als $O(n)$. Dann ist die n -malige Ausführung $O(n^2)$.
- c) Falls $f(n) = \Theta(g(n))$, dann folgt $2^{f(n)} = \Theta(2^{g(n)})$.
- d) $n^n = O(2^n)$
- e) $n = O(n^3)$

Aufgabe 5 **Komplexität von Selection Sort**

Erinnerung: Der *Selection Sort* lautet in Pseudo-Code:

Input: Array $A[0..(n-1)]$ von $n \geq 0$ natürlichen Zahlen

Output: Array A aufsteigend sortiert

SelectionSort(A):

```
for  $j = 0$  to  $n - 2$  {  
     $i = \text{IndexOfMin}(A, j)$ ;  
    if ( $j \neq i$ ) {  
        Swap( $A, i, j$ );  
    }  
}
```

Input: Array $A[0..(n-1)]$ von $n > 0$ natürlichen Zahlen; Startindex j

Output: Index $i \geq j$ des kleinsten Elements im Rest $A[k..(n-1)]$

IndexOfMin(A, j):

```
 $i = j$ ;  
for  $k = j + 1$  to  $n - 1$  {  
    if ( $A[k] < A[i]$ ) {  
         $i = k$ ;  
    }  
}  
return  $i$ ;
```

Input: Array $A[0..(n-1)]$ von $n > 0$ natürlichen Zahlen; Indices i, j

Output: Array A mit Zellen i und j vertauscht

Swap(A, i, j):

```
 $k = A[j]$ ;  
 $A[j] = A[i]$ ;  
 $A[i] = k$ ;
```

- Was ist die exakte Komplexität von **Swap** im RAM-Modell? Wie schreibt man diese in O -Notation?
- Was ist die Laufzeit von **IndexOfMin**? Was ist der *Best Case*, was der *Worst Case*? Geben Sie auch für diese Funktion die Komplexität in O -Notation an!
- Schätzen Sie die Komplexität von **SelectionSort** unter Nutzung der Ergebnisse aus den ersten beiden Teilaufgaben!

Aufgabe 6 **Komplexitätsschätzung**

Schätzen Sie die Komplexitäten der folgenden Algorithmen asymptotisch ab!

a) **rot13**(*string*):

```
encrypted = string;  
for  $i = 0$  to  $\text{size}(\text{string}) - 1$  {  
    if ( $\text{string}[i] \geq \text{'a'} \wedge \text{string}[i] \leq \text{'z'}$ ) {  
         $\text{abstand} = \text{string}[i] - \text{'a'}$ ;  
         $\text{rotierter\_abstand} = (\text{abstand} + 13) \bmod 26$ ;  
         $\text{encrypted}[i] = \text{'a'} + \text{rotierter\_abstand}$ ;  
    }  
}  
  
return encrypted;
```

```
b) MatrixMultiplikation(A, B):  
  if (cols(A)≠rows(B)) {  
    return -∞;  
  }  
  n = cols(A);  
  C = matrix(rows(A), cols(B));  
  for i = 0 to rows(A)-1 {  
    for j = 0 to cols(B)-1 {  
      C[i][j] = 0;  
      for k = 0 to n - 1 {  
        C[i][j] = C[i][j] + A[i][k] · B[k][j];  
      }  
    }  
  }  
  return C;
```